Implementation Report for Applicant Details System

NC1605B: Group Project

Submitted to Module Lead: Dr Zear Ibrahim

Submitted By Upom Shanker Bipro

ID:2447336

Table of Contents

[1. Introduction 2](#_Toc193628737)

[2. Implementation Table 2](#_Toc193628738)

[3. Technical Code Description 3](#_Toc193628739)

[4. Testing 6](#_Toc193628740)

[5. Challenges Faced 7](#_Toc193628741)

[6. Conclusion 7](#_Toc193628742)

[7. References 8](#_Toc193628743)

[8. Appendix 8](#_Toc193628744)

# 1. Introduction

The goal of the project is to develop an Applicant Details System using Java, focusing on viewing, editing, managing, and uploading associated documents and applicant data. The system should enable users to upload profile pictures, personal information, academic qualifications, and documents, all while ensuring smooth functionality, error handling, and data persistence across sessions.

This system is intended to manage applicant records, store data in a structured format, and provide easy access to information for both users and applicants. The core technologies for this implementation include Java Swing for the GUI, with file handling and validation mechanisms built in using Java I/O classes.

# 2. Implementation Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Task Description | Status | Implementation Details |  |
|  | 1. Import Applicant Data | Completed | Method loadApplicantsFromDatabase(); loads data from "APPLICANTS.csv" file. |  |
|  | 2. Implement Data Structures for Efficient Retrieval | Completed | Appropriate data structures have been implemented for storing and retrieving applicant information. |  |
|  | 3. Develop User Interface for Applicant Details | Completed | Implemented using Swing components (JFrame, JPanel, JTextField, etc.) |  |
|  | 4. Profile Picture Upload | Completed | The system accepts PNG files with dimensions not exceeding 300x300 pixels, with error handling for invalid formats. Implemented with validation in openFileChooser() method |  |
|  | 5. Passport Image Upload | Completed | Passport images are uploaded in PNG format with a maximum size of 500x500 pixels. Implemented with validation in openFileChooser() method |  |
|  | 6. Input Academic Qualifications and Grades | Completed | The system allows for input of qualifications with corresponding grades (e.g., "High School Diploma" with grade "A"). Implemented in createQualificationPanel() |  |
|  | 7. Upload Qualification Document | Completed | Users can upload a PDF document as evidence of academic qualifications, with validation before accepting the file. Implemented in createEnhancedCertificateUploadPanel() |  |
|  | 8. Applicant Comment Section | Completed | A comment section has been included for applicants to leave additional notes. Implemented in createCommentsPanel() with comment history functionality |  |
|  | 9. Persistence of Data Across Sessions | Completed | Data, including personal details and uploaded documents, is saved to an appropriate storage format, and the system ensures persistence across sessions. |  |
|  | 10. Search Functionality | Completed | A search function has been implemented to find applicants by their unique ID. Implemented in filterApplicants() with advanced search options |  |
|  | 11. File Upload Feedback | Completed | Clear feedback is provided to users about successful or failed file uploads based on validation. |  |
|  | 12. Error Handling | Completed | Proper error handling ensures users are notified about issues with file uploads, invalid data, or system failures. |  |
|  | 13. Unique ID generation | Completed | Implemented in DatabaseManager.generateApplicantId() method |  |

# 3. Technical Code Description

The Applicant Management System is implemented as a Java Swing application with a SQLite database backend. The implementation follows a modular approach with clear separation of concerns between the user interface, data management, and database operations.

**Core Architecture**

The system is built around two main classes:

ApplicantManagementSystem.java - The main UI class that contains all interface components and event handlers

DatabaseManager.java - A utility class that handles all database operations

**User Interface Implementation**

The UI is designed with a split-pane layout, with the left side displaying a list of applicants and the right side showing detailed information through a tabbed interface. The main UI components are created and configured in the setupUI() method, which establishes the overall layout structure.

**Key UI methods include:**

createPersonalDetailsPanel() - Creates the form for viewing and editing personal information

createDocumentsPanel() - Implements document upload and management

createQualificationPanel() - Manages academic qualification data

createCommentsPanel() - Provides functionality for adding and viewing comments

**Data Management**

The application uses an Applicant class (inner class of ApplicantManagementSystem) to represent applicant data in memory.

Data flow through the application follows this sequence:

1. Initial data is loaded from the database via loadApplicantsFromDatabase()
2. User interactions trigger updates to the in-memory objects
3. Changes are persisted to the database using methods from DatabaseManager

**Document Management**

Document handling is a significant feature of the system. The implementation includes:

* Validation for file types and dimensions in openFileChooser()
* Storage of files in a structured file system under a "documents" directory
* Database references to file paths for efficient retrieval
* Document viewing capability through system default applications

**Search Functionality**

The search implementation in filterApplicants() provides both basic and advanced capabilities:

* Text-based filtering of the applicant list
* Special ID search detection through pattern recognition
* Visual feedback for search results using the flashSearchField() method

**Error Handling**

Error handling is implemented throughout the application with custom dialogs and appropriate feedback:

* Database errors are handled in handleDatabaseError()
* Custom dialogs are created using showCustomDialog()
* Input validation occurs at appropriate points before data processing

**Changes from Original Design**

Compared to the initial Assessment A design, several enhancements were made:

1. Addition of a tabbed interface for better organization of applicant data
2. Implementation of document management with preview capability
3. Enhanced search functionality with special ID search features
4. Improved error handling with custom dialogs
5. More comprehensive form validation for data integrity

**Order of Execution**

The application initialization follows this sequence:

1. Database connection is established in the constructor
2. Database tables are verified and created if needed via initializeDatabase()
3. Existing data is loaded through loadApplicantsFromDatabase()
4. The UI is set up with setupUI()
5. Event handlers are attached to UI components
6. The application enters the event dispatch loop for user interaction

This modular architecture ensures clear separation of concerns and maintainable code structure while fulfilling all the specified requirements for the Applicant Management System.

# Testing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Component | Description | Test Data / Input | Expected Output | Pass/Fail | Correction? |
| 1 | Applicant Management System | **Verifying applicant creation and editing** | Enter an applicant’s name, email, phone number, address, and date of birth. | System created an applicant record in the database with correct details. | Pass | N/A |
| 2 | **Document Upload** | **Verifying file upload with validation** | Upload a PNG profile picture and a PDF certificate. | The file should be uploaded and stored, and the path saved in the database. | Pass | N/A |
| 3 | **Search Functionality** | **Search applicants by name or ID** | Search for applicants using ID or name. | The system returned matching applicants and displayed them in the list. | Pass | N/A |
| 4 | **Error Handling** | **Test invalid file upload (wrong format)** | Try uploading a JPG image when PNG is required. | The system showed an error message saying the file format is invalid. | Pass | N/A |
| 5 | **Database Consistency** | **Test applicant record persistence** | Close the application and reopen it after adding an applicant. | The system should show the previously added applicant with all details intact. | Pass | N/A |

For higher quality Images, check appendix.

# 5. Challenges Faced

- **Image File Size and Format Validation:** I tried my best to ensure that the uploaded images meet the size and format requirements which was the most challenging task for me. Proper validation and error feedback mechanisms had to be implemented to prevent users from uploading unsupported formats or images that exceeded the specified size.

- **Persistent Data Across Sessions:** One of the challenges which I faced was ensuring that the application data persisted between sessions, especially when the GUI was closed and reopened. This involved properly managing file I/O to store and retrieve information in the correct format.

- **Error Handling:** It was crucial to implement comprehensive error handling, especially for file uploads and data parsing. The system needed to handle scenarios like invalid file types or sizes, missing required fields, and incomplete data inputs.

- **GUI Usability:** I ensured that the GUI was user-friendly, intuitive, and easy to navigate was a challenge. Designing the interface to support all required functionalities without overwhelming the user required careful consideration of the layout and interaction flow.

**- Coordination:** Nobody in the group sat for group discussion and were not available until the submission of this report.

# 6. Conclusion

The Applicant Details System has been successfully implemented to meet the requirements of managing applicant data, uploading associated documents, and ensuring data persistence across sessions. The system provides a robust and user-friendly interface that allows users to manage applicant records efficiently, while also ensuring validation and error handling. The challenges faced during the implementation were addressed with appropriate solutions, making the system reliable and functional. The project demonstrates a comprehensive understanding of Java-based GUI development, file handling, and user interaction, providing a solid foundation for future enhancements.

# 7. References

1. Oracle, 2024. Java Platform, Standard Edition Documentation. [online] Available at: <https://docs.oracle.com/javase/8/docs/>.
2. Eclipse Foundation (2024) *Eclipse IDE* (Version 2024-03) [Computer program]. Available at: <https://www.eclipse.org/> (Accessed: 21 March 2025).
3. SQLite (2024) *SQLite* (Version 3.45.2) [Computer program]. Available at: https://www.sqlite.org/index.html (Accessed: 21 March 2025).

# 8. Appendix

**- Source Code:**

Login System:

****

** **

Main Panels:



Database:

 

Main:



Jars:

 

**- Screenshots:**

Login/Signup:

A screen shot of a login

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Main Panel:

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Search Functionality:

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Utilities:

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Errors And Feedbacks:

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.